Optimum Uniform Piecewise Linear Approximation of Planar Curves

JAMES GEORGE DUNHAM, MEMBER, IEEE

Abstract—**Two-dimensional digital curves are often uniformly approximated by polygons or piecewise linear curves. Several algorithms have been proposed in the literature to find such curves. We present an algorithm that finds a piecewise linear curve with the minimal number of segments required to approximate a curve within a uniform error with fixed initial and final points. We compare our optimal algorithm to several suboptimal algorithms with respect to the number of linear segments required in the approximation and the execution time of the algorithm.**

Index Terms—**Optimal approximation, piecewise linear approximation, planar curve segmentation, polygonal approximation, scan-along.**

I. Introduction

PIECEWISE linear approximation of a two-dimensional digital curve is often a compact and effective representation of the curve for shape analysis and pattern classification [1], [2]. Such representations facilitate the extraction of numerical "features" for the description or classification of the given curve.

The problem of piecewise linear approximation for two-dimensional curves has received considerable attention in the literature [3]–[17]. These approximation problems are classified according to the type of norm used in the approximation, to whether the knots of the piecewise linear curve are fixed or variable, and to the continuity or discontinuity of the approximation. We shall consider the case where the knots are constrained to lie on the digital curve to be approximated, the approximations are continuous, and the norm is uniform where the maximum of the absolute value of the pointwise error between the point and the approximating line segment is taken.

Ramer [3] in 1972 and Duda and Hart [4] in 1973 proposed the first such algorithm which iteratively splits the approximation curve into finer and finer linear segments until the error criterion is satisfied on each segment. In 1974, Pavlidis and Horowitz [5] proposed an algorithm based upon splitting and merging segments as well as adjusting segment end points until the desired criterion is satisfied, while Reumann and Witkam [15] proposed a strip algorithm that approximates the tangent to the curve of the fixed endpoint. Williams [6] in 1978 proposed an algorithm based upon the idea of finding the longest segment such that all subsegments with the same initial point satisfy the error criterion. Davis [9] in 1979 proposed an algorithm based upon the smoothed k-curvature of the planar curve. In 1980, Sklansky and Gonzalez [7] proposed an algorithm based upon finding the longest possible segment to satisfy the error criterion. Both Williams, and Sklansky and Gonzalez found scan-along implementations of their algorithms which made them run more efficiently. Badi'i and Peikari [8] in 1982 proposed an algorithm that finds a segment satisfying the error criterion up to a maximal possible length, while Pavlidis [10] developed an algorithm based upon the notion of the points in a line segment being almost collinear. Recently, in 1985, Roberge [11] proposed an enhanced version of Reumann and Witkam's algorithm which was designed to run very quickly.

Since the time for processing features of a polygon is often proportional to the number of linear segments, it is desirable to find approximations with the fewest number of linear segments that satisfy the uniform error criterion. In Section II, we present an algorithm that achieves such an approximation with fixed initial and final points. We call this the optimal algorithm since it seeks a representation with a minimal number of linear segments. The principle of optimality [18] is used and the algorithm is somewhat similar to Bellman's [19] dynamic programming solution of an approximation problem using a squared error norm.

In Section III, an efficient scan-along implementation of the optimal algorithm is developed along the lines of Williams [6] and Sklansky and Gonzalez [7]. We point out that neither Williams nor Sklansky and Gonzalez sufficiently take into account how the distance between a potential endpoint and the intitial point of a segment affects the scan-along procedure. We clarify this point and show its effect on the scan-along procedure.

In Section IV, we compare the optimal algorithm with the algorithms of Ramer [3], Pavlidis and Horowitz [5], Williams [6], Sklansky and Gonzalez [7], Badi'i and Peikari [8], Davis [9], Pavlidis [10], and Roberge [11]. We compare the performance of these algorithms on four closed digital curves with respect to the number of linear segments required in the approximation and the execution time of a Fortran 77 implementation on a TI professional computer. Finally, conclusions are drawn in Section V.

II. Optimal Algorithm

A discrete planar curve parameterized by $0 \leq i \leq I$ is described by

Manuscript received March 25, 1985; revised June 24, 1985.

The author is with the Department of Electrical Engineering, Southern Methodist University, Dallas, TX 75275.

IEEE Log Number 8405793.

0162-8828/86/0100-0067$01.00 © 1986 IEEE

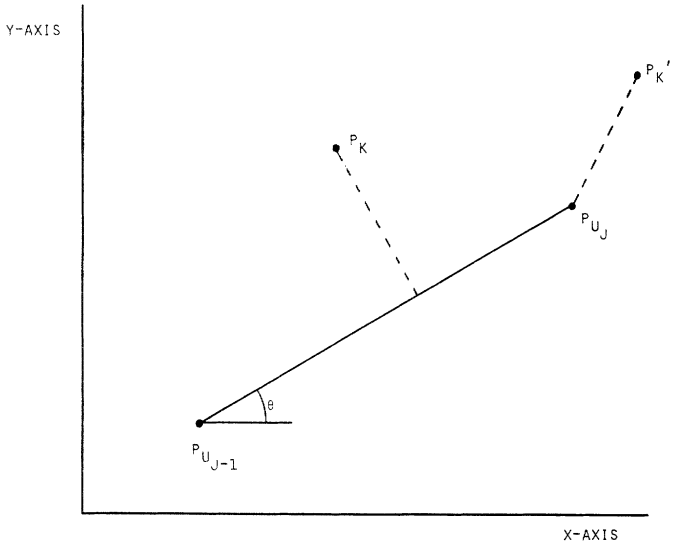


Fig. 1. Computing the approximation error.

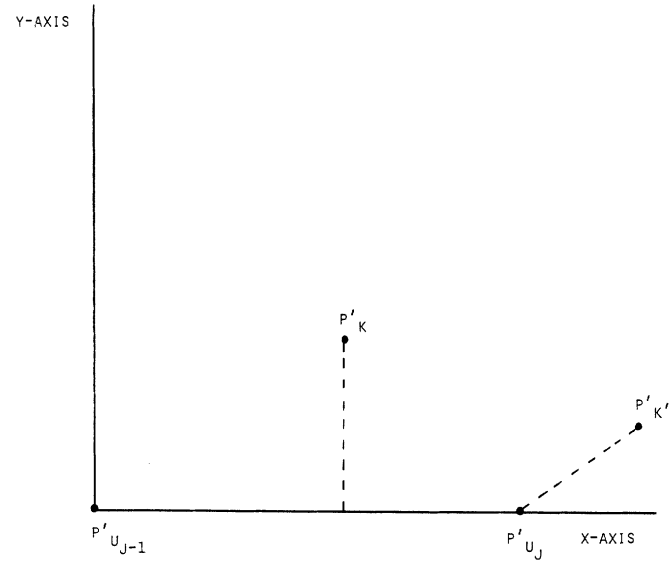


Fig. 2. Translated and rotated axis.

$$C_d = \{(x_i, y_i) | 0 \le i \le I\} .$$

For simplicity, we sometimes denote the point (x_i, y_i) by p_i.

The discrete planar curve is to be approximated by a continuous piecewise linear planar curve C_a whose breakpoints are on C_d and satisfy

$$0 = u_0 < u_1 < u_2 < \cdots < u_N = I.$$

Thus, the jth interval of C_a consists of the straight line that connects the points $p_{u_{j-1}}$ and p_{u_j}, $\overline{p_{u_{j-1}}, p_{u_j}}$, and it approximates the points p_k of C_d where $u_{j-1} \le k \le u_j$.

The error between a point of C_d and the approximating planar curve C_a is the Euclidean distance of the point to the nearest point in the approximation interval. For example, in Fig. 1 the approximation interval is the straight line connecting $p_{u_{j-1}}$ and p_{u_j}, $\overline{p_{u_{j-1}}, p_{u_j}}$. The error between point p_k and the line $\overline{p_{u_{j-1}}, p_{u_j}}$ is found by the line which ends at p_k and is perpendicular to the line $\overline{p_{u_{j-1}}, p_{u_j}}$. The error between the point $p_{k'}$ and the line $\overline{p_{u_{j-1}}, p_{u_j}}$ is the distance of the line between $p_{k'}$ and p_{u_j}, $\overline{p_{k'}, p_{u_j}}$. We denote this error by $e(u_{j-1}, u_j, k)$ and $e(u_{j-1}, u_j, k')$, respectively.

We next give a formal procedure for computing the error between a point and the approximating line segment. If we translate the coordinates so that $p_{u_{j-1}}$ is at the origin of the x–y axis and then rotate by the amount $-\theta$ where θ is the angle between the line $\overline{p_{u_{j-1}}, p_{u_j}}$ and the x-axis of Fig. 1, we get the situation shown in Fig. 2 where all Euclidean distances are preserved. Now if the x-coordinate of p'_k lies between 0 and the x-coordinate of p'_{u_j}, then the error is simply the absolute value of the y-coordinate of p'_k. If the x-coordinate of p'_k is less than 0, then the error is the distance between p'_k and the origin. If the x-coordinate of p'_k is greater than p'_{u_j}, then the error is the distance between p'_k and p'_{u_j}. Using elementary trignometry, we get

$$\cos\theta = \frac{x_{u_j} - x_{u_{j-1}}}{(x_{u_j} - x_{u_{j-1}})^2 + (y_{u_j} - y_{u_{j-1}})^2} \qquad (1a)$$

$$\sin\theta = \frac{y_{u_j} - y_{u_{j-1}}}{(x_{u_j} - x_{u_{j-1}})^2 + (y_{u_j} - y_{u_{j-1}})^2} . \qquad (1b)$$

The coordinate translation and rotation of the point p_k is accomplished by

$$x'_k = (x_k - x_{u_{j-1}}) \cos\theta + (y_k - y_{u_{j-1}}) \sin\theta \qquad (2a)$$

$$y'_k = -(x_k - x_{u_{j-1}}) \sin\theta + (y_k - y_{u_{j-1}}) \cos\theta. \qquad (2b)$$

Thus, for $u_{j-1} \le k \le u_j$ the error is

$$e(u_{j-1}, u_j, k) = \begin{cases} \sqrt{(x_k - x_{u_{j-1}})^2 + (y_k - y_{u_{j-1}})^2} \\ \quad \text{if } x'_k < 0 \\ \sqrt{(x_k - x_{u_j})^2 + (y_k - y_{u_j})^2} \\ \quad \text{if } x'_k > x'_{u_j} \\ |y'_k| \\ \quad \text{else.} \end{cases} \qquad (3)$$

We note that when the transformed x-coordinate of the point lies between the transformed end points of the line, the distance formula is equivalent to that given by Pavlidis and Horowitz [5].

Having defined the error between a point and its approximating line segment, we define the error over a segment as

$$e(u_{j-1}, u_j) = \max_{u_{j-1} \le k \le u_j} e(u_{j-1}, u_j, k).$$

Finally, we define the error between C_d and its approximation C_a as

$$e(C_d, C_a) = \max_{1 \le j \le N} e(u_{j-1}, u_j)$$

and thus we have a uniform norm.

For a given planar curve C_d, fix the maximum allowable approximation error to be ϵ. Our goal is to find a contin-

uous piecewise linear planar curve C_a with breakpoints on C_d and with a minimal number of segments N where $e(C_d, C_a) \leq \epsilon$. Given an index u, let $V(u)$ denote all the points with $v \leq u$ where $e(v, u) \leq \epsilon$ and let $F(u)$ denote the minimal number of segments needed to approximate the curve C_d from points p_0 to p_u within ϵ. Then $F(u)$ satisfies

$$F(u) = \min_{v \in V(u)} 1 + F(v) \tag{4}$$

with initial conditions

$$F(1) = 1 \quad \text{and} \quad F(0) = 0$$

since $u \geq 2$, there is one line segment $\overline{p_v, p_u}$ and then a minimal number of possible segments from 0 to v. This is a particular application of the principle of optimality [18]. We can recursively compute $F(u)$ using dynamic programming with the given initial conditions and $F(I)$ will be the minimal number of segments needed for C_a to approximate C_d within ϵ.

We note that there have been other efforts to implement optimal algorithms based upon dynamic programming. The first work to consider the approximation of curves by line segments using dynamic programming was Bellman [19] in 1961. Gluss wrote a series of papers [20]–[23] expanding upon Bellman's work. Lawson [24] in 1964 used dynamic programming to establish the existence of a balanced error property. Bellman, Gluss, and Roth [25] extended the notion to quasilinearization in 1964. In 1971, Cox [26], apparently unaware of Bellman's work, rediscovered the usefulness of dynamic programming. Vandewalle [27] in 1975 mentions adapting Bellman's solution to the uniform error norm, but gives no details or descriptions of the algorithm.

There are several significant differences between our optimal algorithm and Bellman's algorithm. First, Bellman uses the mean square error or l_2 norm while we use the uniform or l_∞ norm. Second, Bellman considers a discontinuous solution while ours is continuous. Third, from a philosophical point of view, Bellman fixes the number of line segments and minimizes the error while we fix the error and minimize the number of line segments. We note that our recursion is easier to implement and can result in substantially less computational effort, especially for small values of error.

By modifying the recursion slightly, we can also generate the breakpoints for C_a. With the calculations of $F(u)$ store one of the $v \in V(u)$ that achieves the minimum in (4). Then from $F(I) = F(u_N)$ we get u_{N-1}, from $F(u_{N-1})$ we get u_{N-2}, and so forth, thus generating an optimal continuous piecewise linear planar curve that approximates C_d with fixed endpoints u_N and u_0 within error ϵ.

When the planar curve C_d is closed, then the endpoints are the same and so $p_0 = p_I$. In this case the optimal algorithm may not find the approximation C_a with the smallest number of segments. However, we observe that the point p_0 must be contained in some segment of the curve. By applying the optimal algorithm with endpoints $p_1, p_2, \cdots$ until we have exhausted all possible segments $\overline{p_{u_{N-1}}, p_{u_N}}$ containing p_0 with $e(u_{N-1}, u_N) < \epsilon$, one of these will have the overall optimal approximation to the closed curve. In practice, the optimal approximation derived from endpoints $p_0 = p_I$ should tend to be very close to the overall optimal.

III. Scan-Along Implementation

The major difficulty in an efficient implementation of the optimal algorithm is in the evaluation of the set $V(u)$. We now present a scan-along approach for determining $V(u)$.

Let p_0 be the origin of a line segment in Fig. 3. For the first point p_1 draw a circle around it of radius ϵ. Then the set S_1 represents all lines through p_0 which pass within distance ϵ of p_1. For the point p_2 draw a circle around it of radius ϵ and the set S_2 represents all lines through p_0 which pass within ϵ of p_2. Further, the set $T_2 = S_1 \cap S_2$ represents all lines through p_0 which pass within ϵ of both p_1 and p_2. Similarly, for the point p_3 the set $T_3 = S_1 \cap S_2 \cap S_3$ represents all lines through p_0 which pass within ϵ of points p_1, p_2, and p_3.

Mathematically, let

$$\phi_i = \arctan\left(\frac{y_i - y_0}{x_i - x_0}\right) \tag{5}$$

and

$$\psi_i = \arcsin\left(\frac{\epsilon}{(x_i - x_0)^2 + (y_i - y_0)^2}\right). \tag{6}$$

Then

$$S_i = \{\theta \mid \phi_i - \psi_i \leq \theta \leq \phi_i + \psi_i\}$$

and

$$T_i = \bigcap_{j=1}^{i} S_j .$$

We can recursively compute T_i as follows. Let

$$T_i = [\theta_{\min}(i), \theta_{\max}(i)]. \tag{7}$$

Then

$$\theta_{\min}(i+1) = \min((\max(\theta_{\min}(i), \phi_{i+1} - \psi_{i+1})), \theta_{\max}(i) + \delta) \tag{8}$$

$$\theta_{\max}(i+1) = \max((\min(\theta_{\max}(i), \phi_{i+1} + \psi_{i+1})), \theta_{\min}(i) - \delta). \tag{9}$$

If T_i is empty, there can be no line passing within ϵ of points $p_0, p_1, \cdots, p_i$ and, hence, no more points in $V(p_0)$. If T_i is not empty, then if $\phi_i \in T_i$, all the points $p_0, p_1, \cdots, p_i$ are potentially within ϵ of the line $\overline{p_0, p_i}$ while if $\phi_i \notin T_i$, some points in $p_0, p_1, \cdots, p_i$ is greater than ϵ away from the line $\overline{p_0, p_i}$ and $p_i \notin V(p_0)$. For example, in Fig. 3 the points p_0, p_1, p_2 are within ϵ of the line $\overline{p_0, p_2}$ while the point p_2 is more than ϵ away from the line $\overline{p_0, p_3}$.

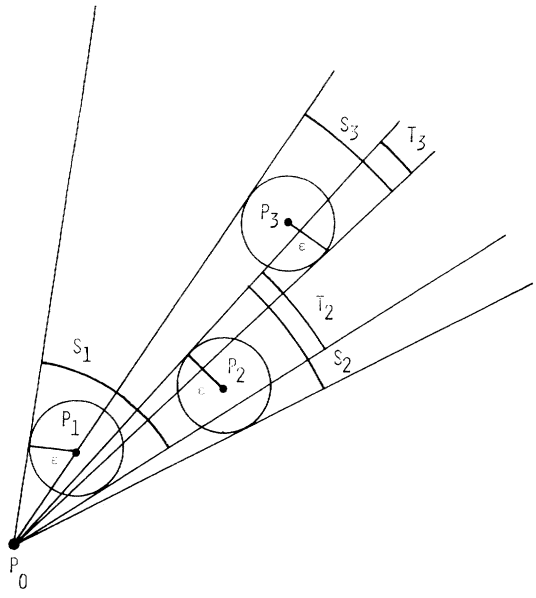


Fig. 3. Scan-along procedure.

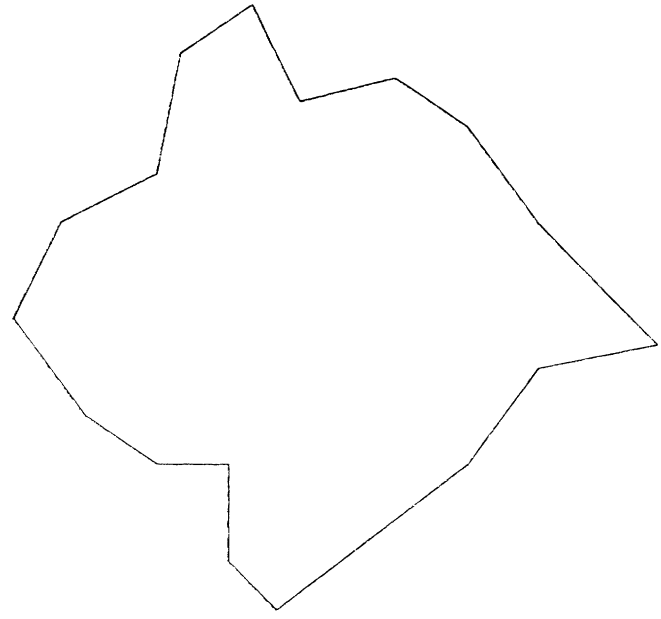


Fig. 4. Small test figure.

Next, we must consider the fact that the line $\overline{p_0, p_i}$ does not go on forever. Let the distance between points p_0 and p_i be

$$d_i = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2}$$

and the maximum distance is

$$d_{\max,i} = \max_{1 \le j \le i} d_j .$$

If $d_{i+1} \ge d_{\max,i}$, then the x_j' coordinate of the translated and rotated point given by (2a & b) will lie between 0 and x_{i+1}' for $0 \le j \le i$. Hence, all the points $p_0, p_1, \cdots, p_i$ are within ϵ of the line $\overline{p_0, p_{i+1}}$ and so $p_{i+1} \in V(p_0)$. If $d_{i+1} < d_{\max,i}$, then we only need to check all the points whose distance is greater than d_{i+1} using (3) to see if they are all within ϵ of the line $\overline{p_0, p_{i+1}}$. If these points are within ϵ of $\overline{p_0, p_{i+1}}$, then $p_{i+1} \in V(p_0)$, otherwise $p_{i+1} \notin V(p_0)$. Finally, if $d_i < \epsilon$, we don't need to worry about this point as it is always within ϵ of any line through p_0.

IV. Experimental Results

In this section we discuss some experimental results. We chose to compare nine algorithms which are divided into two classes. In the first class are algorithms in which the approximation error can be directly controlled: the Ramer [3] and Duda and Hart [4] algorithm using the error measure (3); the Pavlidis and Horowitz [5] algorithm where the R algorithm for adjusting segment endpoints is taken from Pavlidis [28] with $M = 1$; the Williams [6] algorithm using exact calculations and taking into account distance as discussed at the end of Section III; the Sklansky and Gonzalez [7] algorithm taking into account distance discussed at the end of Section III; the Badi'i and Peikari [8] algorithm neglecting the parts of the algorithm associated with the l_1 norm and using maximal extension lengths of 5, 10, 15, 20, 25, 30, 40, and 50; and the optimal algorithm discussed in Section II and III. The second class of algorithms consists of those in which the approximation error is indirectly controlled by some parameter of the algorithm: the Davis [9] algorithm where the smoothed k-curvature was increased in increments of 1 until the maximum allowable error was surpassed; the Pavlidis [10] algorithm where the shaded region of Fig. 12.5 of [10] was determined by the equation

$$T = 0.4583 + 0.25\ C$$

and k_0 took on the values 1, 2, 3, 4, 6, 8, and 10; and the Roberge [11] algorithm where the extension factor took on the values 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, and 4.0.

Four closed curves were chosen to compare the algorithms with regard to the number of vertices in the approximation and the total time to process the planar curve. Fig. 4 is a small polygon with 20 vertices, Fig. 5 is the outline of a chromosome taken from Montanari [12] with 148 vertices, Fig. 6 is the outline of a right lung consisting of 402 vertices, and Fig. 7 is the outine of Lake Lewisville near Dallas, Texas, and has 527 vertices. All algorithms find continuous approximations with breakpoints along C_d with the same initial and final vertices for allowable errors of 0.2, 0.4, 0.6, 0.8, 1.0, 1.5, 2.0, 2.5, 3.0, and 4.0. All algorithms were implemented in Fortran 77 and executed on a TI professional computer with an 8087 numeric coprocessor for floating point calculations. All algorithms consider the planar curve points as real numbers, input data in the same way, and write out results using the same formats. For those algorithms where the approximation error was indirectly controlled or there were several options for a fixed error, we first chose a result with a minimal number of vertices to satisfy a given approximation error and then, among all those with the same number of vertices, chose one with the minimal program execution time.

Tables I–IV show the number of vertices required in the approximation of Figs. 4–7, respectively. Several obser-

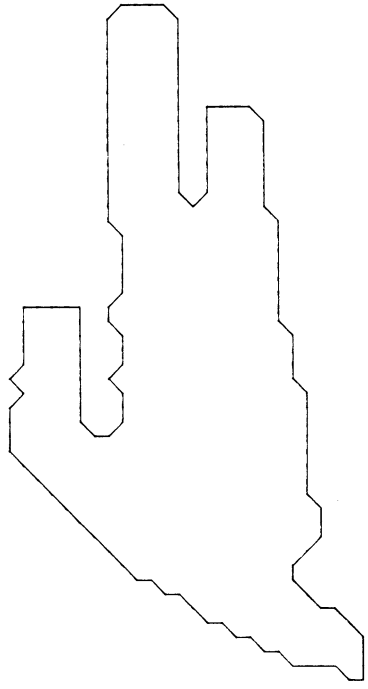


Fig. 5. Chromosome outline taken from Montanari [9].

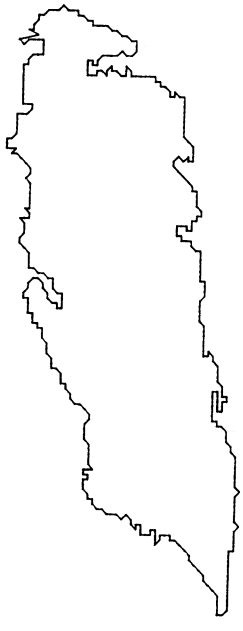


Fig. 6. Right lung figure.

vations can be made from the data. First, the optimal algorithm always required the least number of vertices as expected. Considering the suboptimal algorithms in which the approximation error is directly controlled, we see that the Sklansky and Gonzalez algorithm tended to outperform all the rest followed by the Badi'i and Peikari algorithm. When the error was small, the Ramer, and Pavlidis and Horowitz algorithms performed about the same followed by the Williams algorithm. When the error was large the Williams algorithm tended to outperform the Pavlidis and Horowitz algorithm, which in turn tended to outperform the Ramer algorithm. For the suboptimal algorithms

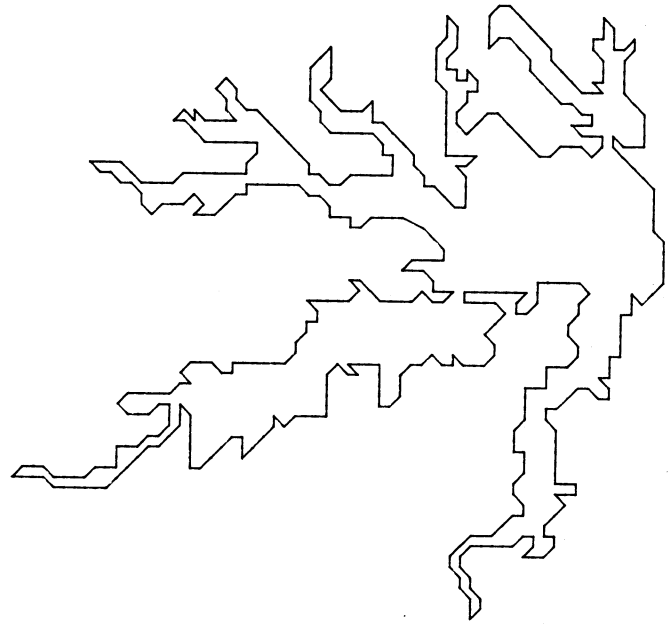


Fig. 7. Lake Lewisville near Dallas, TX.

in which the approximation error is indirectly controlled, for small errors the Pavlidis algorithm tended to outperform the Roberge algorithm while for large errors the Roberge algorithm tended to outperform the Davis algorithm, which in turn tended to outperform the Pavlidis algorithm. Comparing the two classes of algorithms, the algorithms with direct control over the error tended to uniformly outperform the algorithms with indirect control over the error.

Second, for the very small errors all algorithms tended to perform the same. The improvement offered by the optimal algorithm was greatest over the middle range of error. Finally, all algorithms performed about the same on the small polygonal figure, but showed distinct differences on the three larger figures.

Tables V–VIII show the total execution time for the algorithms for Figs. 4–7, respectively, in seconds. Several observations can be made. First, for the Ramer, Davis, and Badi'i and Peikari algorithms, the time tended to decrease slightly as the error increased while for the Pavlidis algorithm the time tended to increase slightly. The time for the Pavlidis and Horowitz algorithm and the Roberge algorithm tended to increase to a maximum then decrease as the error increased. The time for both the Williams, and Sklansky and Gonzalez algorithms tended to first decrease to a minimum and then increase as the error increased. For the optimal algorithm, the running time increased as the error increased which is expected since increasing the error increased the size of $V(u)$.

Second, the Ramer, Williams, Sklansky and Gonzalez, Davis, Pavlidis, and Roberge algorithms tended to have a running time proportional to the total number of vertices in the curve. This tended to be true for the Pavlidis and Horowitz algorithm and the Badi'i and Peikari algorithm and was definitely not true for the optimal algorithm.

Third, there was little difference in the running time of

TABLE I
NUMBER OF VERTICES FOR SMALL FIGURE

Error	Ramer	Pavlidis Horowitz	Williams	Sklansky Gonzalez	Badi'i Peikari	Davis	Pavlidis	Roberge	Optimal
0.2	18	18	18	18	18		18	18	18
0.4	18	18	18	18	18		18	18	18
0.6	17	17	17	17	17		17	18	17
0.8	17	15	15	15	15		17	16	15
1.0	15	14	14	14	14		14	15	14
1.5	11	10	12	10	10		14	12	10
2.0	7	9	9	7	7		10	12	7
2.5	6	8	8	6	6	7	6	10	6
3.0	5	5	5	5	5	5	5	5	5
4.0	5	5	5	5	5	5	5	5	5

TABLE II
NUMBER OF VERTICES FOR CHROMOSOME

Error	Ramer	Pavlidis Horowitz	Williams	Sklansky Gonzalez	Badi'i Peikari	Davis	Pavlidis	Roberge	Optimal
0.2	59	59	59	59	59		59	59	59
0.4	51	54	59	48	52		59	59	46
0.6	36	39	43	32	37		46	59	30
0.8	29	30	32	23	26		37	40	23
1.0	19	21	17	16	19		26	37	16
1.5	16	17	16	16	16		22	34	16
2.0	16	15	11	12	13	27	22	15	11
2.5	13	13	10	9	11	26	18	14	9
3.0	11	10	9	8	10	20	18	10	8
4.0	8	9	9	7	8	17	9	9	7

TABLE III
NUMBER OF VERTICES FOR LUNG

Error	Ramer	Pavlidis Horowitz	Williams	Sklansky Gonzalez	Badi'i Peikari	Davis	Pavlidis	Roberge	Optimal
0.2	254	254	254	254	254		254	254	254
0.4	228	226	254	215	224		254	254	207
0.6	159	160	185	137	157		193	222	133
0.8	116	116	124	95	116		155	222	86
1.0	92	79	87	63	93		153	161	56
1.5	56	51	51	37	50		129	70	35
2.0	40	33	36	26	35	105	129	51	24
2.5	34	26	27	22	27	93	110	47	18
3.0	31	26	22	17	26	80	91	39	15
4.0	20	19	18	12	18	78	53	22	10

TABLE IV
NUMBER OF VERTICES FOR LAKE LEWISVILLE

Error	Ramer	Pavlidis Horowitz	Williams	Sklansky Gonzalez	Badi'i Peikari	Davis	Pavlidis	Roberge	Optimal
0.2	330	330	330	330	330		330	330	330
0.4	291	298	329	274	293		329	330	266
0.6	229	211	227	198	219		329	330	192
0.8	168	174	175	140	157		219	330	130
1.0	145	134	134	105	143		190	247	101
1.5	100	87	84	72	95		177	190	63
2.0	71	71	67	54	64	135	152	131	48
2.5	60	49	50	40	50	113	152	80	39
3.0	51	40	43	36	45	102	152	67	28
4.0	37	28	31	22	30	93	151	37	21

TABLE V
PROGRAM EXECUTION TIME FOR SMALL FIGURE IN SECONDS

Error	Ramer	Pavlidis Horowitz	Williams	Sklansky Gonzalez	Badi'i Peikari	Davis	Pavlidis	Roberge	Optimal
0.2	9.0	12.0	8.9	9.3	10.3		10.5	7.8	9.6
0.4	9.3	12.2	8.9	9.4	10.3		10.5	7.8	9.8
0.6	9.3	12.2	8.8	9.3	10.1		10.1	7.8	9.8
0.8	9.3	12.3	8.8	9.2	10.0		10.1	8.1	9.8
1.0	9.2	12.6	8.7	9.2	9.9		9.8	8.5	9.8
1.5	9.0	12.7	8.6	9.1	9.4		9.8	8.9	10.0
2.0	8.8	12.6	8.5	8.9	9.1		10.1	8.9	10.2
2.5	8.8	12.7	8.5	8.8	9.0	8.6	10.3	8.0	10.3
3.0	8.7	11.4	8.3	8.8	9.0	8.2	10.2	8.1	10.4
4.0	8.8	11.4	8.3	8.9	9.0	8.2	10.2	8.1	10.6

TABLE VI
PROGRAM EXECUTION TIME FOR CHROMOSOME IN SECONDS

Error	Ramer	Pavlidis Horowitz	Williams	Sklansky Gonzalez	Badi'i Peikari	Davis	Pavlidis	Roberge	Optimal
0.2	16.4	83.4	16.2	16.7	24.3		15.1	13.1	24.2
0.4	16.2	84.6	16.2	17.1	19.0		15.1	13.1	27.0
0.6	15.7	80.1	15.8	16.8	19.6		17.3	13.1	34.1
0.8	15.5	73.6	15.6	16.5	17.5		16.8	13.8	41.5
1.0	15.2	66.4	15.2	15.9	17.8		18.8	14.0	44.3
1.5	15.0	66.3	15.1	16.2	23.1		19.5	13.9	49.4
2.0	15.0	58.5	15.1	16.2	18.2	14.4	19.5	13.8	59.1
2.5	14.9	58.4	15.2	16.2	19.4	14.4	16.4	13.4	67.7
3.0	14.8	55.4	15.4	16.7	20.2	14.1	16.4	12.5	74.3
4.0	14.7	54.5	16.3	17.8	18.5	14.4	15.1	13.2	105.8

TABLE VII
PROGRAM EXECUTION TIME FOR LUNG IN SECONDS

Error	Ramer	Pavlidis Horowitz	Williams	Sklansky Gonzalez	Badi'i Peikari	Davis	Pavlidis	Roberge	Optimal
0.2	35.0	213.3	34.8	36.0	76.2		26.7	25.6	45.0
0.4	33.9	230.5	34.5	36.1	69.1		26.7	28.7	50.2
0.6	31.9	223.5	32.8	35.3	95.0		32.5	28.7	63.8
0.8	30.6	230.8	31.3	34.5	112.2		32.3	26.6	80.2
1.0	30.1	241.1	30.1	32.9	47.4		35.4	26.6	96.8
1.5	28.1	224.9	29.7	32.4	51.9		31.8	23.5	132.6
2.0	28.3	199.3	30.7	33.8	47.3	26.4	31.8	23.9	182.1
2.5	27.7	175.8	31.1	35.2	31.5	26.3	32.0	23.4	253.0
3.0	27.6	176.5	31.9	36.2	31.2	26.0	42.1	22.2	340.9
4.0	26.9	151.0	37.9	42.4	29.9	25.9	34.4	22.2	522.8

TABLE VIII
PROGRAM EXECUTION TIME FOR LAKE LEWISVILLE IN SECONDS

Error	Ramer	Pavlidis Horowitz	Williams	Sklansky Gonzalez	Badi'i Peikari	Davis	Pavlidis	Roberge	Optimal
0.2	45.4	334.3	44.3	45.8	70.3		34.3	32.3	55.8
0.4	44.4	318.7	44.5	46.6	65.0		34.3	32.3	63.9
0.6	41.9	329.8	41.1	45.0	54.3		34.3	32.3	73.9
0.8	39.4	312.6	39.0	42.5	74.0		40.0	32.3	86.6
1.0	39.3	322.2	38.4	42.0	56.3		52.6	33.5	98.8
1.5	37.6	335.6	38.1	43.3	53.8		39.4	33.2	137.3
2.0	36.6	308.6	39.2	44.8	65.8	32.4	39.5	31.2	178.4
2.5	36.1	312.8	41.7	48.1	49.1	32.4	39.5	31.3	227.3
3.0	36.0	282.2	43.5	51.3	46.0	32.2	39.5	29.8	298.7
4.0	34.7	284.6	49.3	60.9	43.1	32.0	82.9	29.1	488.1

the Ramer and Williams algorithm for most errors. Fourth, the running time of the optimal algorithm was comparable to that of the Ramer, Williams, Pavlidis, and Sklansky and Gonzalez algorithms for small errors. Fifth, the running time for the small polygonal figure was about the same for all nine algorithms. Sixth, the Roberge had the fastest running time.

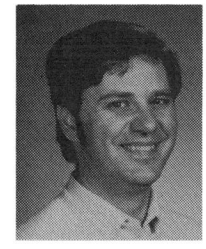
V. CONCLUSION

An algorithm for optimal uniform piecewise linear approximation of planar curves was presented for fixed initial and final vertices. A scan-along procedure was described to effectively implement the algorithm. The performance of the optimal algorithm was compared to eight other algorithms with regard to the number of vertices required and the total program execution time. Of all the algorithms compared, the Roberge [11] algorithm consistently had the shortest program execution time, but tended to require twice as many vertices as the optimal algorithm. Of the eight suboptimal algorithms, the Sklansky and Gonzalez [7] algorithm, taking into account distance as discussed at the end of Section III, consistently required the smallest number of vertices to meet a given error tolerance and was relatively close to the number of vertices required by the optimal algorithm. The optimal algorithm outperformed the Sklansky and Gonzalez algorithm with regard to the number of vertices required in the approximation, but took more time. For small errors the time of both algorithms was comparable.

REFERENCES

[1] T. Pavlidis, "A review of algorithms for shape analysis," *Comput. Graphics Image Processing*, vol. 7, pp. 243–258, 1978.
[2] —, "Algorithms for shape analysis of contours and waveforms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-2, pp. 301–312, 1980.
[3] U. Ramer, "An iterative procedure for the polygonal approximation

of plane curves," *Comput. Graphics Image Processing*, vol. 1, pp. 244–256, 1972.
[4] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973, pp. 328–339.
[5] T. Pavlidis and S. L. Horowitz, "Segmentation of plane curves," *IEEE Trans. Comput.*, vol. C-23, pp. 860–870, Aug. 1974.
[6] C. M. Williams, "An efficient algorithm for the piecewise linear approximation of planar curvers," *Comput. Graphics Image Processing*, vol. 8, pp. 286–293, 1978.
[7] J. Sklansky and V. Gonzalez, "Fast polygonal approximation of digitized curves," *Pattern Recognition*, vol. 12, pp. 327–331, 1980.
[8] F. Badi'i and B. Peikari, "Functional approximation of planar curves via adaptive segmentation," *Int. J. Syst. Sci.*, vol. 13, no. 6, pp. 667–674, 1982.
[9] L. S. Davis, "Shape matching using relaxation techniques," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-1, pp. 60–72, Jan. 1979.
[10] T. Pavlidis, *Algorithms for Graphics and Image Processing*. New York: Computer Science, 1982, pp. 281–297.
[11] J. Roberge, "A data reduction algorithm for planar curves," *Comput. Vision, Graphics, Image Processing*, vol. 29, pp. 168–195, 1985.
[12] U. Montanari, "A note on minimal length polygonal approximation to a digitized contour," *Commun. ACM*, vol. 13, pp. 41–47, Jan. 1970.
[13] J. Sklansky, R. L. Chazin, and B. J. Hansen, "Minimum perimeter polygons of digitized silhouettes," *IEEE Trans. Comput.*, vol. C-21, pp. 260–268, Mar. 1972.
[14] I. Tomek, "Two algorithms for piecewise linear continuous approximations of functions of one variable," *IEEE Trans. Comput.*, vol. C-23, pp. 445–448, Apr. 1974.
[15] K. Reumann and A. P. M. Witkam, "Optimizing curve segmentation in computer graphics," in *International Computer Symposium*, A. Gunther, B. Levrat, and H. Lipps, Eds. New York: Elsevier, 1974, pp. 467–472.
[16] K. Ichida and T. Kiyono, "Segmentation of planar curves," *Electron. Commun. Japan*, vol. 58-D, no. 11, pp. 689–696, 1975.
Press, 1957.
[17] Y. Kurozumi and W. A. Davis, "Polygonal approximation by the minmax method," *Comput. Graphics Image Processing*, vol. 19, pp. 248–264, 1982.
[18] R. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton Univ. Press, 1957.
[19] R. Bellman, "On the approximation of curves by line segments using dynamic programming," *Commun. ACM*, vol. 4, p. 284, 1961.
[20] B. Gluss, "Further remarks on line segment curve-fitting using dynamic programming," *Commun. ACM*, vol. 5, pp. 441–443, Aug. 1962.
[21] —, "A line segment curve-fitting algorithm related to optimal encoding of information," *Inform. Contr.*, vol. 5, pp. 261–267, 1962.
[22] —, "Least squares fitting of planes to surfaces using dynamic programming," *Commun. ACM*, vol. 6, pp. 172–175, Apr. 1963.
[23] —, "An alternative method for continuous line segment curve-fitting," *Inform. Contr.*, vol. 7, pp. 200–206, 1964.
[24] C. L. Lawson, "Characteristic properties of the segmented rational minimax approximation problem," *Numer. Math.*, vol. 6, pp. 293–301, 1964.
[25] R. Bellman, B. Gluss, and R. Roth, "On the identification of systems and the unscrambling of data: Some problems suggested by neurophysiology," *P. NAS US*, vol. 52, pp. 1239–1240, 1964.
[26] M. G. Cox, "Curve fitting with piecewise polynomials," *J. Inst. Math. Appl.*, vol. 8, pp. 36–52, 1971.
[27] J. Vandewalle, "On the calculation of the piecewise linear approximation to a discrete function," *IEEE Trans. Comput.*, vol. C-24, pp. 843–846, 1975.
[28] T. Pavlidis, "Waveform segmentation through functional approximation," *IEEE Trans. Comput.*, vol. C-22, pp. 689–697, July 1973.



James George Dunham (S'72–M'77) was born in Akron, OH, on September 10, 1950. He received the B.S., M.S., and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, in 1973, 1973, and 1978, respectively.

From August of 1977 to August of 1984 he was with the Department of Electrical Engineering, Washington University, St. Louis, MO, where he was an Associate Professor. Since August of 1984 he has been an Associate Professor of Electrical Engineering at Southern Methodist University in Dallas, TX. His research interests and activities are in the areas of communications, information theory, and cryptography.

Dr. Dunham is a member of Tau Beta Pi and Phi Beta Kappa.